Name: Zhuodong Huang
Student ID: 28261551
Game: Block Battle
Language: Java

Block Battle Report

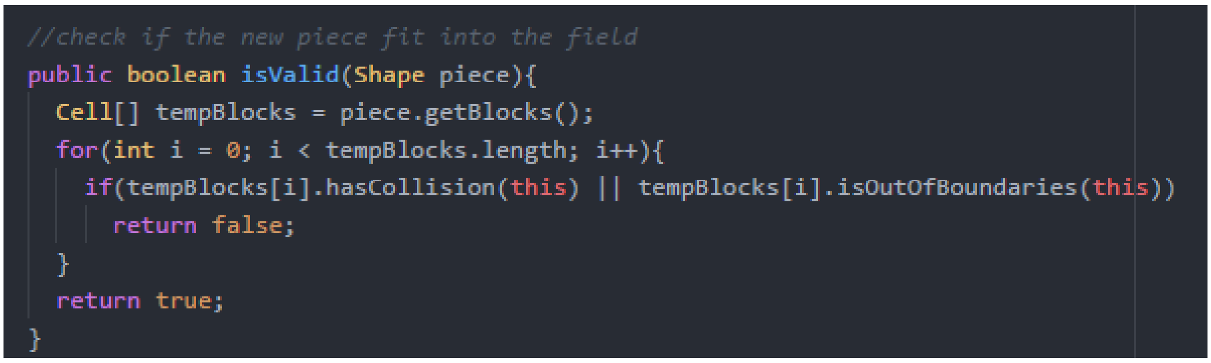
Introduction:

Block Battle is one of the famous games in AI field. In this game, there are two main problems, one is how to decide the position of the piece. The other one is how to evaluate the reward. Each piece has different shape and size, for example, I, L, T, J, S, Z, O types and 2*2, 3*3, 4*4 sizes. Position is represent the left top corner of the piece, however not every piece is start from the first column. For example, if we make a right turn for "I" shape, it will start at the 3rd column. So we will need to create a helper method to check if the position is valid. Also, different ways to evaluate the reward, it will also provide different solutions. We will need to think about how to define a hole in different case. The hole in different state should have different punishment. The deeper hole it has, the larger punishment it get. The purpose for this game is that we will need to develop an AI agent which can decide the next action(Left, Right, Turn) in order to win the game. It is important to choose the good strategy. After I watch and analysis the games in the Theaigame website, I think that the score we earn during the game is not the most important part of the game. Instead, making the block as low as I can is the key to win. I see a lot of bots have higher score but lost the game due to it reaches the top. Also, according to the rule, every fifteen round, it will get a solid row, which means as the time pass, it will become more and more easier to reach the top. Therefore, the strategy that I choose is to clean the row each round. For this strategy, agent will focus on the current piece, and find if any position can clean a row. If it exist a row clean, the agent will more prefer to choose that as action. If not, holes inside the block will become a huge obstacle, so in this case, bot should choose the action that makes less holes in order to get a better state.

Strategy detail and methods:

After I try different kinds of programming language several times, I finally decide to use Java, which the one I most familiar with. The first strategy I try is like genetic algorithm. I randomly put piece on the top of the block and use the fitness function to find the best one. This strategy takes long time. Also, since it is random, sometimes it doesn't pick the good position. My final strategy is similar to reinforcement learning, the AI will get reward and punish from hole, clean line. It will search all the position from left to right that can fit into our field and calculate the reward or punish for each position. Since it will check all the possible position, it won't miss any position like what random pick does.

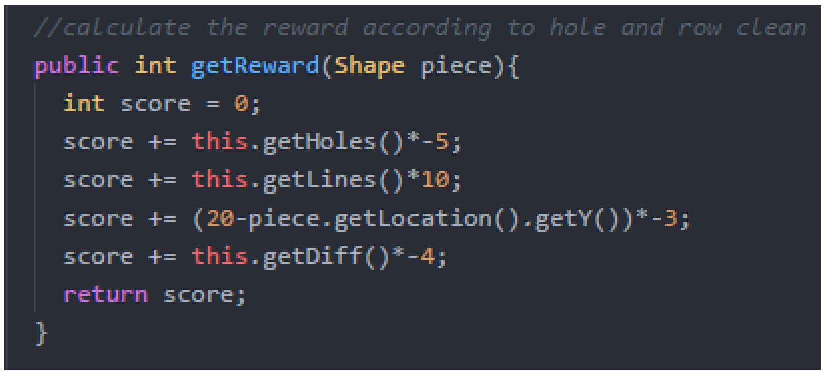
First thing I do is to move the piece all the way to the left most, drop it and evaluate the reward. Record the reward. And then move one right, drop it and evaluate the reward, keep doing this until the piece reach the right side of the field. Fortunately, Java starter code already provide functions, oneLeft() and oneRight(). I can use them to move the tempPiece to the position that I want. However, one thing I need to check is to make sure it won't move outbound of the field. I write a function that call the helper function hasCollision() and isOutOfBoundaries() to check if the piece is inside the field, also fit in this position. Below is what the function looks like:



```
//check if the new piece fit into the field
public boolean isValid(Shape piece){
  Cell[] tempBlocks = piece.getBlocks();
  for(int i = 0; i < tempBlocks.length; i++){
    if(tempBlocks[i].hasCollision(this) || tempBlocks[i].isOutOfBoundaries(this))
      return false;
  }
  return true;
}
```

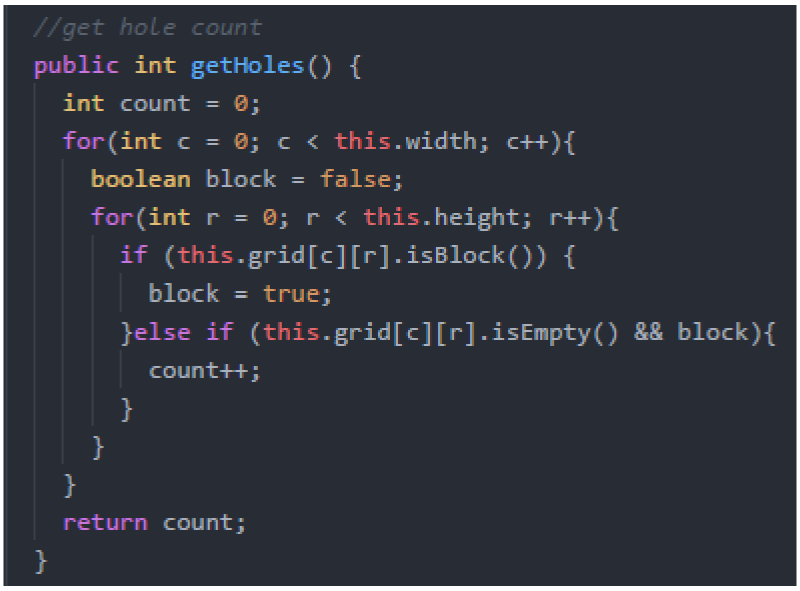
After I done with this condition, I turn the piece and do the same thing as above. There are ten columns and four turns for each piece, so the bot needs to calculate forty different situations. Only the position that has the best reward will be store and return.

One of the most important parts of reinforcement learning is to evaluate the reward. It will collect the data in current state to evaluate the reward with the parameter. I write a function called getReward().



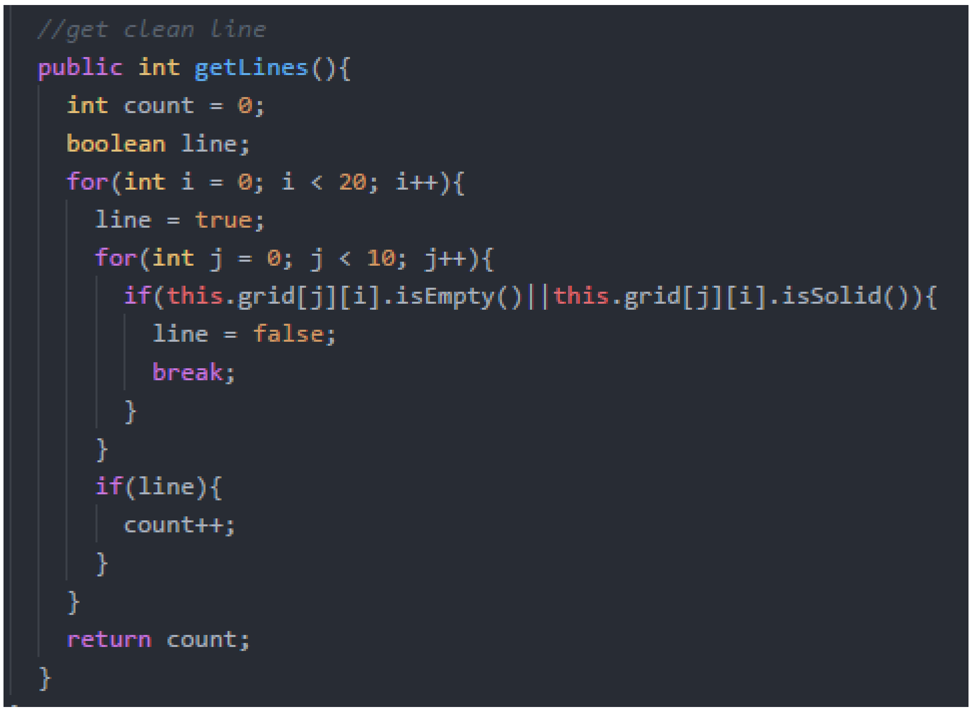
```
//calculate the reward according to hole and row clean
public int getReward(Shape piece){
  int score = 0;
  score += this.getHoles()*-5;
  score += this.getLines()*10;
  score += (20-piece.getLocation().getY())*-3;
  score += this.getDiff()*-4;
  return score;
}
```

This function will consider three things, holes, clean lines and height. For the holes, it will get punishment depend on how many hole exist. In this way, agent will try to avoid getting holes inside the block.



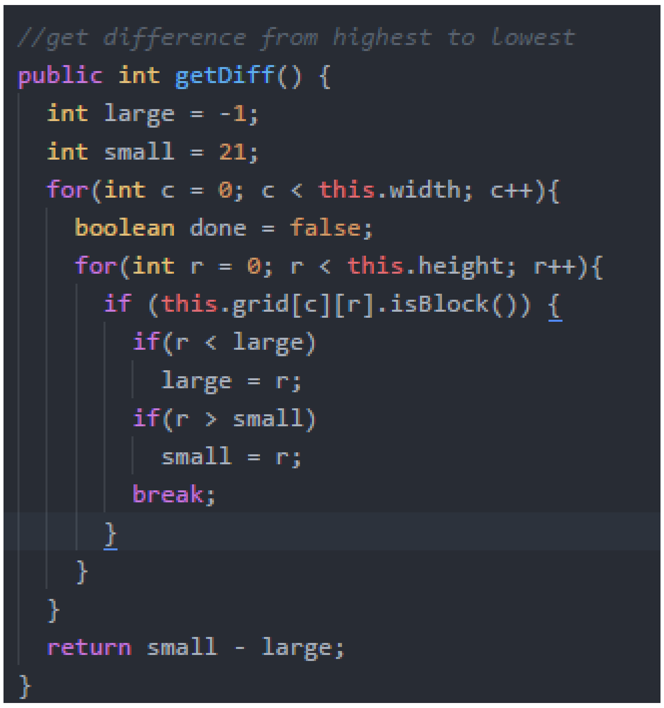
```
//get hole count
public int getHoles() {
  int count = 0;
  for(int c = 0; c < this.width; c++){
    boolean block = false;
    for(int r = 0; r < this.height; r++){
      if (this.grid[c][r].isBlock()) {
        block = true;
      }else if (this.grid[c][r].isEmpty() && block){
        count++;
      }
    }
  }
  return count;
}
```

The second thing is clean line, which is an important way to keep the block low. Therefore, if AI agent can clean line, it will get a large reward. Especially, if combo happen, this can make block lower, at the same time, it will also create a garbage lines to your opponent.



```
//get clean line
public int getLines(){
  int count = 0;
  boolean line;
  for(int i = 0; i < 20; i++){
    line = true;
    for(int j = 0; j < 10; j++){
      if(this.grid[j][i].isEmpty()||this.grid[j][i].isSolid()){
        line = false;
        break;
      }
    }
    if(line){
      count++;
    }
  }
  return count;
}
```

The last one is height. Try to keep each columns to have the same height will slow down the height of block increase. Therefore, I keep tracking the difference of higher bound and lower bound.



```
//get difference from highest to lowest
public int getDiff() {
  int large = -1;
  int small = 21;
  for(int c = 0; c < this.width; c++){
    boolean done = false;
    for(int r = 0; r < this.height; r++){
      if (this.grid[c][r].isBlock()) {
        if(r < large)
          large = r;
        if(r > small)
          small = r;
        break;
      }
    }
  }
  return small - large;
}
```

Above reward function will return the current reward for the current state. It will compare with the best score. The one which has better score will store into array will the position move it has.

The last thing I need to do is to use the data in the array(index 0 is reward, index 1 is move and index 2 is turn) to determine what is the action for the agent.

Similarly to the pervious project we done in the course, the parameters for the reward in reinforcement learning have a huge effect to our result. So I need to do a huge amount of training, and use the data from the output to adjust the reward parameter. Keep adjusting the parameter until the agent work well enough.

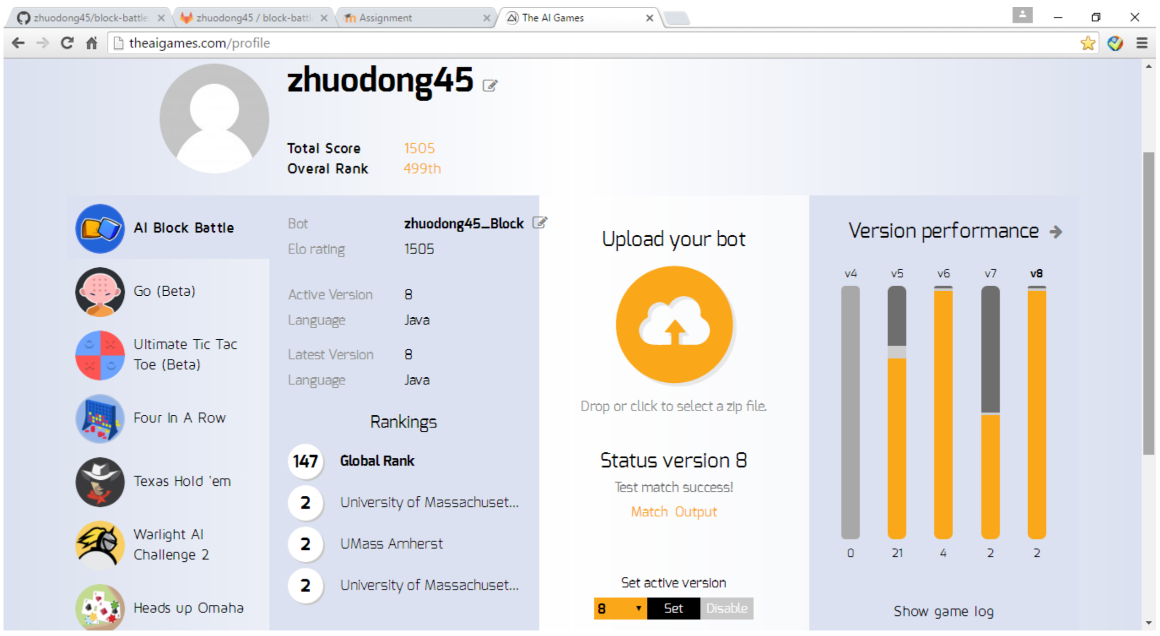
Result:

The reinforcement learning is a good approach for this block battle game. It works well on my bot. basically what is does is the bot collects the current state and it predict the future state. Using those states to evaluate the reward or punishment and depends on the reward to decide the next action. However, this approach takes a long time to train my bot and adjust the parameter. We only have about 2 weeks to implement our bot, which in this short amount time, the bot may not train to the best condition. Other than that, genetic algorithm is also a good way to implement this game. Genetic algorithm is much easy to implement and also it takes less time to train my bot. In a long term, I believe reinforcement learning is a better approach for this game. Also, after I try both strategies, reinforcement learning seems working better on my bot. Therefore, I choose reinforcement learning.

Some old version of my bot use genetic algorithm, it is easy to implement. My bot uses the fitness function to determine the best move. Later on, the latest version of my bot is apply reinforcement learning. I can see some improve for my bot. I also add some new reward function to help my bot evaluate the state more precise. After training it online with other bots, and adjust the parameter. My bot is getting better and better. Especially, I add another function for my bot to predict the future piece, my bot can last at least ten more round for each run.

Currently, my bot in University of Massachusetts leaderboard is rank 2, which is a good grade, and the global rank is about 147. GitLab link and screenshot will show below:

GitLab: https://gitlab.com/zhuodong45/block-battle
Ranking:



zhuodong45

Total Score 1505
Overal Rank 499th

AI Block Battle
Go (Beta)
Ultimate Tic Tac Toe (Beta)
Four In A Row
Texas Hold 'em
Warlight AI Challenge 2
Heads up Omaha

Bot	zhuodong45_Block
Elo rating	1505
Active Version	8
Language	Java
Latest Version	8
Language	Java

Rankings

147	Global Rank
2	University of Massachuset...
2	UMass Amherst
2	University of Massachuset...

Upload your bot

Drop or click to select a zip file.

Status version 8
Test match success!
Match Output

Set active version
8 Set Disable

Version performance

v4	v5	v6	v7	v8
0	21	4	2	2

Show game log

Discussion and conclusions:

I got a lot of fun by implementing this block battle game. When I see my bot getting better, I am proud of that. Remake my strategy, the first try is genetic algorithm. I implement some fitness function for my bot, which is according to the holes and line clean. My bot is working, but it still has a lot of bugs and it doesn't work good enough. The second try is reinforcement learning. Fortunately, reward functions are similar to fitness function. It is easy for me to write reward function for reinforcement learning. Each piece has four turns. For each turn, I use search algorithm from left to right, top to bottom, to find the right position. Apply the reward function to get the reward. Finally decide the next action according the best reward.

At the beginning, my reward function only considers the hole and line clean for current piece. To improve my bot, I then add more reward functions for it. I let the bot also look ahead to consider the next coming piece and the height of the block. These make a huge improve for my bot. My rank increases about 30.

However, as I mention before, reinforcement learning need a lot of training in order to adjust the parameter more precise. My latest version bot doesn't have too much training with other bots online. Therefore, the reward parameter is not pretty good. I believe in this limit time, my bot is still not perfect. It also has a lot of space can be improved. For example, in my strategy, I don't care too much about the score I earn during the game. The score doesn’t affect my bot, but it is a way to push my opponent into a danger state. In the future, I can add another reward function that considers the score in the game. Another way to improve my bot is to use T-spin, which is mention in theaigame’s website. T-spin will make the turn at the end of the moves, this can clean some lines that are hard to clean.

Lastly, for this block battle game, I try genetic algorithm and reinforcement learning on it. Both work well, even thought my bot is not perfect. My reward function only considers holes, clean lines and height, which is not completely consider all the factor in the state. In the future, when I have time, I will try to improve my reward function and adjust its parameter. Also, I would like to try to implement deep learning, which one of the most famous approach for machine learning. This approach takes much longer to training our bot, but it is one of the great approaches. I don’t have too much knowledge with it. I believe I will come back and try it after I finish machine learning course.